Proceedings of the
46th IEEE Conference on Decision and Control
New Orleans, LA, USA, Dec. 12-14, 2007

A Variational Approach to Constructivist Learning for Mobile Robot Navigation

Tejas R. Mehta and Magnus Egerstedt

Abstract— **In this paper, we present a constructivist approach for the *Learning by Example* problem, where control laws (or behaviors) are learned in order to approximate a training trajectory. The new behaviors are learned by systematically improving upon existing capabilities. Within this context, the learning problem is formulated as an optimal control problem, and variational arguments are used to obtain optimality conditions. Numerical algorithms that utilize the optimality conditions to attain a stationary solution are produced. A small-scale navigation example is discussed in order to highlight the operation of the proposed approach.**

I. Introduction

Consider a situation in which a human operator is driving a robot to a specified goal location through an unknown environment. One would typically expect the human operator to try to find safe paths to the goal while avoiding hazardous regions. Moreover, it is conceivable that the robot (through human control) reacts to distinctive features in the environment in a particular way, such as "stay at least 1 meter away from the wall". A natural objective is to have the robot mimic the actions of the human operator in a completely autonomous manner. However, the problem is not to simply store the path that the human-operated robot took and then reproduce it, but rather to *learn* at a behavioral level the control laws, i.e., closed-loop mappings from sensory input data to control signals, needed to reproduce this motion. We will refer to this problem as the *Learning From Example* problem, and variants of it have received considerable attention in the robotics community [1], [2], [3], [4], [5], [6], [7], [8], [9].

Many different strategies for the Learning From Example problem have emerged over the years. Generally speaking, these strategies can be placed in two major research camps. One camp approaches the problem from a perception point of view and attempts to learn the relevant features (e.g., paths, walls, etc). Then these features are classified as traversable or non-traversable based on the learning cues from the observed human behavior. In other words, what is learned is a feature classification that can serve as guidance for the robot to plan through the terrain [3], [4], [10]. We will refer to this as the *perception-centric* approach. This approach can be complemented with the control-based view, where behaviors that closely resemble the motion demonstrated by the human operator are learned. Typically, the relevant features are assumed to be *a priori* known, and what is learned is a policy mapping features to control signals [11], [12], [13]. This *control-centric* approach will be taken in this paper.

A common feature among previous control-centric methods is that they try to learn behavioral mappings using a "blank-slate" view, i.e., they attempt to learn these mappings without reference to previously established capabilities [1], [2], [14], [15], [16]. However, a natural modification of this approach would consist of a systematic improvement of the existing capabilities. Consider for example the scenario of learning to ride a motorcycle. Assuming we already know how to ride a bicycle, we will not completely throw out this knowledge when learning to ride a motorcycle. In fact, we use our experience from riding a bicycle to leverage the learning of riding a motorcycle. Indeed, *constructivism* views learning as a process in which the learner actively constructs or builds new ideas or concepts based upon current and past knowledge [17], [18].

We take this point-of-view throughout this paper and approach the Learning From Example problem by constructing new control laws from previously established ones. In particular, we assume that we have access to a set of behaviors (possibly of limited capabilities). It is natural to consider such previous capabilities as designed with a particular task in mind such as "avoiding-obstacles" or "following-wall." However, no such interpretation is necessary. Within this context, constructivist learning can be viewed as learning new control laws as some function of the known behaviors, where the learning is guided by training examples.

In particular, given a training trajectory, the learning task consists of finding an appropriate sequence of new behaviors in order to approximate the training trajectory. Here, the new behaviors will be defined as linear combinations of the existing behaviors. Thus, the main task involves finding the weights of each existing behavior in the linear combination, and in determining how many such new behavioral combinations are required. We will let the transitions between new behaviors be temporally driven, even though event-driven transitions may be better suited for mobile robot navigation. But, as this work represents an initial study of constructivist learning from an optimal control point of view, we leave this issue to future endeavors.

This constructivist framework for learning draws inspiration from our previous work dealing with optimal control of multi-modal systems [19], [20], [21], where we combined recurring mode string fragments (i.e., a sequence of behaviors that occurs frequently throughout previous runs) into smooth

This work was supported by DARPA through Grant number FA8650-04-C-7131

T. R. Mehta is with the School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA 30332, USA tmehta@ece.gatech.edu

M. Egerstedt is with the School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA 30332, USA magnus@ece.gatech.edu

"meta-modes." In this paper, we apply a similar mode fusion methodology to the Learning From Example problem. The main difference in this paper, besides from the application domain, is the fact that we are now learning a sequence of new behaviors to approximate the training trajectory instead of learning a single new behavior. Since we are dealing with switching between new learned behaviors, the problem also involves optimizing over the switching instants.

The outline of the paper is as follows: In Section II, we formalize the Learning From Example problem as an optimal control problem. The optimality conditions are derived in Section III. In Section IV, we specify numerical algorithms that utilize the optimality conditions to converge to a stationary solution. A simple example, demonstrating the viability of the proposed method, is introduced in Section V, followed by conclusions in Section VI.

II. Problem Formulation

We start by formally introducing the Learning from Example problem and casting it as an optimal control problem. More specifically, we start by assuming a prior collection of behaviors and try to approximate the training trajectory from a combination of these existing behaviors.

Formally, let X denote the state space, Y denote the observation space, and U be the control space. Suppose the system dynamics are

$$\dot{x}(t) = f(x(t), u(t)), \tag{1}$$
$$y(t) = h(x(t)). \tag{2}$$

A behavior is a mapping from observations to control values, i.e., $\kappa : Y \to U$. Hence, if we drive the robot according to behavior κ_1 until time τ_1 and κ_2 for time τ_2, and so on, then the evolution of the system is

$$\dot{x} = \begin{cases} f(x, \kappa_1(y)) & \text{when } t \in [\tau_0, \tau_1) \\ f(x, \kappa_2(y)) & \text{when } t \in [\tau_1, \tau_2) \\ \vdots & \vdots \end{cases} \tag{3}$$

Note that an event-driven version of this model can be defined, where the switching times are driven by interrupts $\xi_i : Y \to \{0, 1\}$. In this case, the switching time τ_i would be given by

$$\tau_i = \min_{t > \tau_{i-1}} \{t \ : \ \xi_i(y(t)) = 1\}. \tag{4}$$

Note that this corresponds to the formalism developed within the Motion Description Language (MDL) framework [22].

Suppose we have a set of behaviors $K = \{\kappa_1, \kappa_2, \ldots, \kappa_N\}$. In the constructivist framework of this paper, the new learned behavior will be defined through a combination of the behaviors in K. One option is to let the new behavior κ_n be given as a linear combination of the existing behaviors. However, in order to learn a richer class of behaviors, we propose the following construction of the new behavior:

$$\kappa_n(y) = \sum_{i=1}^{N} \mu_i(y, \alpha_i)\kappa_i(y), \tag{5}$$

where $\mu_i : Y \times \mathbb{R}^k \to \mathbb{R}$ is a weighing function that is parameterized by control vector $\alpha_i \in \mathbb{R}^k$. We will refer to these weighing functions as *membership functions* as they loosely resemble membership functions in fuzzy logic control [23]. It is easy to see that this more general specification of the new control mode can accommodate the linear combination solution.

Now, given an observed trajectory from the human operated training example, which we denote as $y : [0, T] \to Y$, we are interested in learning a sequence of new behaviors that will approximate the training trajectory. Assume that we know the initial state $x_0 = x(0)$. Further assume that the human operator used M number of mode (note that we will not enforce this assumption in the presented method), then define an approximation trajectory $\tilde{x}(t)$ as follows:

$$\dot{\tilde{x}}(t) = \begin{cases} f(\tilde{x}, \kappa_{n1}(\tilde{y})) & \text{when } t \in [\tau_0, \tau_1) \\ f(\tilde{x}, \kappa_{n2}(\tilde{y})) & \text{when } t \in [\tau_1, \tau_2) \\ \vdots & \vdots \\ f(\tilde{x}, \kappa_{nM}(\tilde{y})) & \text{when } t \in [\tau_{M-1}, \tau_M] \end{cases} \tag{6}$$

with $\tilde{x}(0) = x(0)$. Here the observations $\tilde{y}(t) = h(\tilde{x}(t))$, and the new behavior κ_{nj} is in the for of 5, where α_i^j is the control vector parameterizing membership function μ_i for control law κ_{nj}. The Learning From Example problem can be posed as an optimization problem: choose the control variables α_i^j for $j = 1, \ldots, M$ and $i = 1, \ldots, N$, and the switching times τ_i for $i = 1, \ldots, M-1$ such that the performance criterion

$$J = \int_{\tau_0}^{\tau_M} L(y(t), \tilde{y}(t))dt + \psi(y(\tau_M), \tilde{y}(\tau_M)) \tag{7}$$

is minimized, where $L : Y \times Y \to \mathbb{R}$ is the instantaneous cost, and $\psi : Y \times Y \to \mathbb{R}$ is the terminal cost. Also, in order to utilize the variational methods presented later, we assume that L and ψ are twice differentiable in their second argument. Note here that $\tau_0 = 0$ and $\tau_M = T$ are assumed fixed.

Of course, we do not know the exact number of modes used by the human operator. Hence, we will provide an algorithm for determining the number of modes necessary to approximate the observed trajectories. We will call this algorithm the *outer algorithm*, while the *inner algorithm* will find the optimal control vectors α_i^j (for $i = 1, \ldots, N$ and $j = 1, \ldots, M$) and the optimal switching times τ_i (for $i = 1, \ldots, M-1$) given the number of switches. In order to facilitate this inner algorithm, we will derive the optimality conditions for minimizing the cost (7) given a fixed number of switches in the following section.

III. Optimality Conditions

In this section, we utilize the calculus of variations to derive the optimality conditions for control parameters α_i^j that shape the membership functions μ_i for behavior κ_{nj}, for $i = 1, \ldots, N$ and $j = 1, \ldots, M$, and the optimal switching times $(\tau_1, \ldots, \tau_{M-1})$ with respect to the performance criterion (7) assuming the approximation trajectory has M modes. The approximation trajectory in this case is given by (6). The

central theme in utilizing variational arguments is to adjoin the cost with the constraint, which in our case is given by (6) and (5), via a co-state (or lagrange multiplier) $\lambda(t)$. The main idea here is to perturb the control parameters and compute the Gateaux (or directional) derivative of the cost J in the direction of the perturbation to gain access to the optimality conditions. Since we will be differentiating the cost function, we must make some mild assumptions about differentiability. Namely, assume that L and ψ are continuously differentiable in their second argument.

As mentioned earlier, we obtain the unperturbed cost, denoted by $\tilde{J}_0$, by adding the constraint via a co-state $\lambda(t)$ to (7). For ease of notation, we start by defining the Hamiltonian H_i as

$$H_i(x,\tilde{x},\lambda_i,\vec{\alpha}_i) = L(h(x),h(\tilde{x})) + \lambda_i \tilde{f}_i(\tilde{x},\vec{\alpha}_i), \tag{8}$$

where $\tilde{f}_i(\tilde{x},\vec{\alpha}_i) = f\big(\tilde{x},\kappa_{ni}(h(\tilde{x}))\big)$ and $\vec{\alpha}_i = [\alpha_1^i,\ldots,\alpha_N^i]^T \in \mathbb{R}^{Nk}$. Now, the augmented (but unaltered from an evaluation point of view) unperturbed cost is given by

$$\tilde{J}_0 = \sum_{i=1}^{M}\int_{\tau_{i-1}}^{\tau_i}\Big[H_i(x,\tilde{x},\lambda_i,\vec{\alpha}_i) - \lambda_i\dot{z}\Big]dt + \\ +\psi(y(\tau_M),\tilde{y}(\tau_M)). \tag{9}$$

Now, we perturb (9) in such a way that $\vec{\alpha}_i \to \vec{\alpha}_i + \epsilon\vec{\gamma}_i^{l_r}$, where $\vec{\gamma}_i^{l_r} = [0,\ldots,\gamma_i^{l_r},\ldots,0]^T$ (note the $(kl+r)^{th}$ entry is $\gamma_i^{l_r}$ and all other entries are 0, i.e., we are perturbing the r^{th} entry of shaping vector α_l) and $\tau_i = \tau_i + \epsilon\theta_i$ for $i = 1,\ldots,M-1$, and $\epsilon << 1$, then $\tilde{x} \to \tilde{x} + \epsilon\eta$ is the resulting variation in $\tilde{x}(t)$. The corresponding variation in the observation $\tilde{y}$ is given as follows

$$\tilde{y} \to h(\tilde{x}+\epsilon\eta) = h(\tilde{x}) + \frac{\partial h}{\partial\tilde{x}}(\tilde{x})\epsilon\eta + o(\epsilon)$$

Note that $\tau_0 = 0$ and $\tau_M = T$ are assumed fixed. Thus, the perturbed cost, denoted by $\tilde{J}_\epsilon$, is given by

$$\tilde{J}_\epsilon = \sum_{i=1}^{M}\int_{\tau_{i-1}+\epsilon\theta_{i-1}}^{\tau_i+\epsilon\theta_i}\Big[H_i(x,\tilde{x}+\epsilon\eta,\lambda_i,\vec{\alpha}_i+\epsilon\vec{\gamma}_i^{l_r}) - \\ -\lambda_i\dot{\tilde{x}} - \epsilon\lambda_i\dot{\eta}\Big]dt + \Big[\psi(y,\tilde{y}+\frac{\partial h}{\partial\tilde{x}}(\tilde{x})\epsilon\eta)\Big]_{t=\tau_M} + o(\epsilon). \tag{10}$$

Using a first order approximation, we get

$$\tilde{J}_\epsilon = \sum_{i=1}^{M}\int_{\tau_{i-1}+\epsilon\theta_{i-1}}^{\tau_i+\epsilon\theta_i}\Big[H_i(x,\tilde{x},\lambda_i,\vec{\alpha}_i) + \\ +\frac{\partial H_i}{\partial\tilde{x}}(x,\tilde{x},\lambda_i,\vec{\alpha}_i)\epsilon\eta + \\ +\frac{\partial H_i}{\partial\alpha_i^{l_r}}(x,\tilde{x},\lambda_i,\vec{\alpha}_i)\epsilon\gamma_i^{l_r} - \lambda_i\dot{\tilde{x}} - \epsilon\lambda_i\dot{\eta}\Big]dt + \\ +\Big[\psi(y,\tilde{y}) + \frac{\partial\psi}{\partial\tilde{y}}(y,\tilde{y})\frac{\partial h}{\partial\tilde{x}}(\tilde{x})\epsilon\eta\Big]_{t=\tau_M} + o(\epsilon). \tag{11}$$

Hence, the first order variation in the performance index (7) can be expressed as the limit for $\epsilon \to 0$ of

$$\delta J(\vec{\alpha_1},\ldots,\vec{\alpha_M},\vec{\tau_i};\vec{\gamma}_1^{l_r},\ldots,\vec{\gamma}_M^{l_r},\vec{\theta}) = \lim_{\epsilon\to 0}\frac{\tilde{J}_\epsilon - \tilde{J}}{\epsilon}.$$

Using (9) and (11), it follows that

$$\delta J = \sum_{i=1}^{M}\int_{\tau_{i-1}+\epsilon\theta_{i-1}}^{\tau_i}\Big[\frac{\partial H_i}{\partial\tilde{x}}\eta + \frac{\partial H_i}{\partial\alpha_i^{l_r}}\gamma_i^{l_r} - \lambda_i\dot{\eta}\Big]dt + \\ + \sum_{i=1}^{M-1}\theta_i\Big[\lambda_{i+1}(\tau_i+)\big(\tilde{f}_i(\tau_i-) - \tilde{f}_{i+1}(\tau_i+)\big)\Big] + \\ +\Big[\frac{\partial\psi}{\partial\tilde{y}}\frac{\partial h}{\partial\tilde{x}}\eta\Big]_{t=\tau_M}. \tag{12}$$

Note here that we dropped the arguments in the differentials for the sake of compactness, however they can be readily inferred from (11). The integral terms in (12), denoted by $\delta\mathcal{K}$, can be further reduced by integrating $\lambda_i\dot{\eta}$ by parts to obtain

$$\delta\mathcal{K} = \sum_{i=1}^{M}\int_{\tau_{i-1}+\epsilon\theta_{i-1}}^{\tau_i}\Big[\frac{\partial H_i}{\partial\tilde{x}} + \dot{\lambda_i}\Big]\eta dt - \\ -\sum_{i=1}^{M}\Big[\lambda_i\eta\Big]_{\tau_{i-1}+\epsilon\theta_{i-1}}^{\tau_i} + \sum_{i=1}^{M}\gamma_i^{l_r}\int_{\tau_{i-1}+\epsilon\theta_{i-1}}^{\tau_i}\frac{\partial H_i}{\partial\alpha_i^{l_r}}dt. \tag{13}$$

Recall that $\theta_0 = 0$ since $\tau_0 = 0$ is fixed, and note that $\eta(0) = 0$ since $\tilde{x}(0) = x(0) = x_0$. Now using the fact that $\eta(t)$ is continuous, (13) is further reduced. Substituting $\delta\mathcal{K}$ back into δJ, we note that we can use single continuous co-state $\lambda(t)$ given by

$$\lambda(\tau_M) = \Big[\frac{\partial\psi}{\partial\tilde{y}}(y,\tilde{y})\frac{\partial h}{\partial\tilde{x}}(\tilde{x})\Big]_{t=\tau_M}, \tag{14}$$

$$\lambda(\tau_i-) = \lambda(\tau_i+) \text{ for } i = 1,\ldots,M-1, \text{ and} \tag{15}$$

$$\dot{\lambda}(t) = -\frac{\partial H_i}{\partial\tilde{x}}(x,\tilde{x},\lambda_i,\vec{\alpha}_i) \text{ when } t\in(\tau_{i-1},\tau_i). \tag{16}$$

With this choice of the co-state $\lambda(t)$, which can be solved by integrating (16) backwards with boundary conditions (14)-(15), we obtain

$$\delta J = \sum_{i=1}^{M}\gamma_i^{l_r}\int_{\tau_{i-1}}^{\tau_i}\frac{\partial\tilde{f}_i}{\partial\alpha_i^{l_r}}(\tilde{x},\vec{\alpha}_i)dt + \\ + \sum_{i=1}^{M-1}\theta_i\Big[\lambda(\tau_i)\big(f_i(\tau_i-) - f_{i+1}(\tau_i+)\big)\Big]. \tag{17}$$

Since the variations (θ_i) and $(\gamma_i^{l_r})$ are independent, the necessary conditions for optimality (i.e., $\delta J = 0$) are

$$\frac{\partial J}{\partial\tau_i} = \lambda(\tau_i)\big[\tilde{f}_i(\tau_i-) - \tilde{f}_{i+1}(\tau_{i+1}+)\big] \equiv 0 \\ \text{for } i = 1,\ldots,M-1 \text{ and,} \tag{18}$$

$$\frac{\partial J}{\partial\alpha_i^{l_r}} = \int_{\tau_{i-1}}^{\tau_i}\lambda\frac{\partial\tilde{f}_i}{\partial\alpha_i^{l_r}}(\tilde{x},\vec{\alpha}_i)\ dt \equiv 0 \text{ for } i = 1,\ldots,M, \\ l = 1,\ldots,N, \text{ and } r = 1,\ldots,k. \tag{19}$$

IV. Numerical Algorithms

In the previous section, we derived the optimality conditions for minimizing (7) given a fixed number of modes. In this section, we first present a numerical algorithm that utilizes these optimality conditions to converge to a stationary solution for optimal switching times and shaping parameters.

We call this the *inner algorithm*, which is complemented by the *outer algorithm* that increments the number of modes and weighs the benefit of adding additional modes. The idea here is to start with an approximation trajectory using a single mode and add modes to the approximation trajectory as long as it is beneficial to do so.

The inner algorithm, which employs a gradient descent method, is given as follows:

- Initialize with a guess of the control variables $\vec{\tau}^{(0)}$ and $\vec{\alpha}_i^{(0)}$'s for $i = 1, \ldots, M$, and let $p = 0$.
- `while` $p < 1$ `or` $(J^{(p)} - J^{(p-1)}) < \epsilon$
  - Compute the approximation function $\tilde{x}(t)$, observation $\tilde{y}(t)$, and cost $J^{(p)}$ forward in time from 0 to T using (6), (5), and (7).
  - Compute the co-state $\lambda(t)$ backward in time from T to 0 using (14) and (15).
  - Compute the gradients $\nabla \tilde{J}(\vec{\tau}^{(p)})$, and $\nabla \tilde{J}(\vec{\alpha}_i^{(p)})$ for $i = 1, \ldots, M$.
  - Update the control variables as follow :

$$\vec{\tau}^{(p+1)} = \vec{\alpha}^{(p)} - \gamma^{(p)} \nabla \tilde{J}(\vec{\tau}^{(p)}),$$
$$\vec{\alpha}_i^{(p+1)} = \vec{\alpha}_i^{(p)} - \gamma^{(p)} \nabla \tilde{J}(\vec{\alpha}_i^{(p)}),$$

    for $i = 1, \ldots, M$.
  - $p = p + 1$
- `end while`

Note that the choice of the step-size $\gamma^{(p)}$ can be critical for the method to converge. An efficient method among others is the use of Armijo step-size presented in [24]. Because of the non-convex nature of the cost function J, this gradient descent algorithm will only converge to a local minimum. Hence the attainment of a "good" local minimum can be quite dependent on the choice of a "good" initial guess for the control variables. The association of such a local method with heuristic strategies in order to find a global minimum is not investigated here.

The inner algorithm, outlined above, provides the optimal control variables $\vec{\alpha}_i$'s and optimal switching times τ_i's given a fixed number of modes. However, recall that we do not know the number of modes a priori. Thus, we propose an *outer algorithm* to figure out the number of modes necessary to approximate the observed trajectory. The main idea here is to start by assuming that the observed trajectory can be approximated with a single mode, then continue to increment the number modes as long as there is a "sufficient" reduction in cost. Let this sufficient reduction be encoded by the parameter ρ, then the outer algorithm can be specified as follows:

- Initialize with $k = 1$
- `while` $k < 2$ `or` $(J^*_{(k)} - J^*_{(k-1)}) < \rho$
  - Obtain $J^*_{(k)}$ using the inner algorithm with number of modes $M = k$.
  - $k = k + 1$
- `end while`

The parameter ρ is the thresh hold that weighs the benefit of the reduction in cost J^* versus the increase in complexity introduced by the adding an additional mode in the approximation trajectory. The choice of an appropriate ρ may be critical to convergence of the algorithm, as choosing a small value for ρ may cause the number of modes to increase indefinitely.

V. Examples

A. Simulated Navigation

In this section, we introduce a simple example to demonstrate the viability of the proposed method. Consider the task of navigating a unicycle from a known initial configuration to a specified goal location. The unicycle dynamics are given as

$$\begin{aligned} \dot{x}_1 &= v\cos(x_3), \\ \dot{x}_2 &= v\sin(x_3), \\ \dot{x}_3 &= \omega. \end{aligned} \tag{20}$$

In the system above, (x_1, x_2) is the Cartesian coordinates of the center of the unicycle and x_3 is its orientation with respect to the x_1-axis. The initial configuration of the robot is given as x_0, while the goal x_g is specified as (x_{g_1}, x_{g_2}). The observed trajectory from a training run (solid line), along with the initial configuration and desired goal, are shown in Figure 1. The training trajectory is generated using a sequence of three modes, hence we should be able to approximate it using three modes using the proposed algorithms. We assume that the linear velocity v is fixed, thus the control consists of the angular velocity control term. Also, we assume that we have a state observer, i.e., $y(t) = x(t)$. Now, we wish to learn the number of modes needed to approximate this trajectory as well as a description of the individual modes.

As mentioned earlier, we will start of with a set of previously established behaviors. For this example, we start with two known behaviors, namely "go-to-goal" and "avoid-obstacles." The feedback law corresponding to each of these behaviors is given as

$$\kappa_g(x) = \omega_g = C_g(\phi_g - x_3), \tag{21}$$
$$\kappa_o(x) = \omega_o = C_o(\pi + \phi_o - x_3). \tag{22}$$

Note here that C_g and C_o are the gains associated with each behavior, and ϕ_g and ϕ_o are the angles to the goal and nearest obstacle, respectively. Both of these angles are measured with respect to the x_1-axis and can be expressed as

$$\phi_g = \arctan(\frac{x_{g_2} - x_2}{x_{g_1} - x_1}) \text{ and} \tag{23}$$
$$\phi_o = \arctan(\frac{x_{obs_2} - x_2}{x_{obs_1} - x_1}), \tag{24}$$

where (x_{g_1}, x_{g_2}) and (x_{obs_1}, x_{obs_2}) are the Cartesian coordinates of the goal and the nearest obstacle, respectively. Moreover, the new behaviors will be given by the linear combination of these known behaviors:

$$\kappa_n(x) = \alpha_g \kappa_g(x) + \alpha_o \kappa_o(x). \tag{25}$$

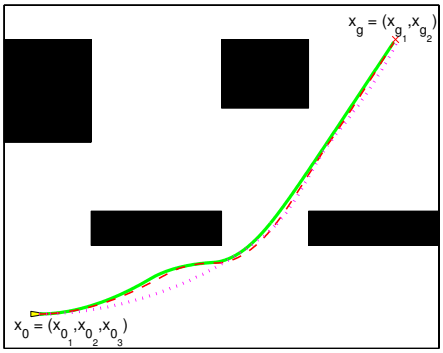


Fig. 1. Depicted is the approximation trajectory ($\tilde{x}$) obtained by using three modes (dashed) and two modes (dotted) along with the original observed trajectory (solid).

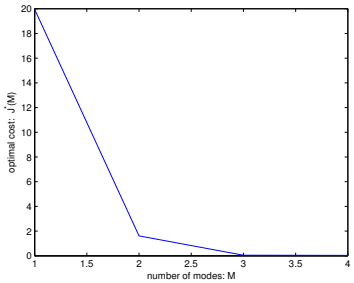
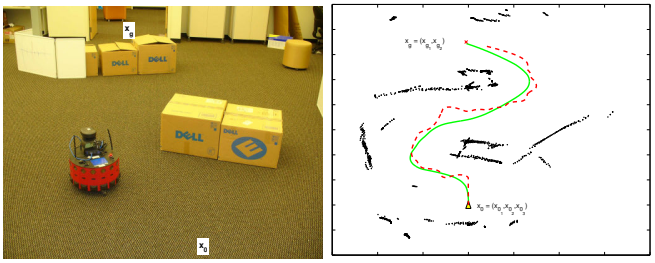


Fig. 2. The optimal cost as a function of the number of modes given by the outer algorithm.

As outlined earlier, we start by attempting to approximate the trajectory with one mode and then increment the number of modes as necessary. For our simulations, the cost is given by $L(y(t), \tilde{y}(t)) = 0.05||\tilde{y}(t) - y(t)||^2$ and $\psi(y(\tau_M), \tilde{y}(\tau_M)) = 10||\tilde{y}(\tau_M) - y(\tau_M)||^2$.

Also, the linear velocity $v = 1$ m/s, the gains $C_g = C_o = 1$, and the thresh hold in the outer algorithm is set to $\rho = 0.1$. The step-size in the inner algorithm can be chosen using the Armijo algorithm. The optimum cost J^* as a function of the number of modes is shown in Figure 2. Since we are using a descent algorithm, the optimal cost refers to a local optimum as a function of the modes. Even though, the figure shows a continuous descent in the optimal cost as the modes increase, note that the optimal cost is only defined when the number of modes is a positive integer.

Observe that the outer algorithm quickly terminates. We deduce that the observed trajectory can be approximated by using three modes, as expected. Note that the cost, in this case, approaches zero as the number of modes grows. This can normally be expected with the limiting case being high frequency switching between modes (i.e., chattering). The goal of our algorithm is to choose a suitable terminating condition (or rho) so that we avoid approaching this limit. The resulting trajectory from using three modes (dashed) and two modes (dotted) along with the original observed trajectory is depicted in Figure 1.



(a) (b)

Fig. 3. (a)The experimental setup using the Magellan Pro robot, (b) Depicted is the approximation trajectory ($\tilde{x}$) obtained by using the learned behavior (dashed) along with the original observed trajectory (solid).

B. Navigation Using the Magellan Pro

The simulated navigation example, presented above, illustrated the operation of the proposed method for the Learning From Example problem. In this section, we will take the promising simulation results to obtain effective navigation strategies for the Magellan Pro robot from iRobot from training runs. The training data was obtained from a joystick operated run guided by a human operator. The training data consisted of the state of the robot (i.e., $[x_1, x_2, x_3]$) and the range sensor readings from the entire run. The experimental setup is shown in Figure 3.

The kinematics of the Magellan Pro can be effectively captured by a unicycle model (20). Moreover, since the experiment is conducted on a carpeted floor, slippage can be ignored. As before, we start out by assuming two known behaviors, namely, "go-to-goal" and "avoid-obstacles." However, for this experiment, we let the new behaviors take on the more general form:

$$\kappa_n(x) = \mu_g(x)\kappa_g(x) + \mu_o(x)\kappa_o(x), \tag{26}$$

where the membership functions are defined as

$$\mu_g(x, \alpha_g) = 1 - e^{-\alpha_g \|x - x_o\|^2}, \tag{27}$$

$$\mu_o(x, \alpha_o) = e^{-\alpha_o \|x - x_o\|^2}, \tag{28}$$

as done in [21].

As outlined earlier, we start by attempting to approximate the trajectory with one mode and then increment the number of modes as necessary. For this experiment, the cost will be as defined in the previous section. It turns out that the training trajectory can be effectively approximated using one new mode (i.e., additional modes did not significantly reduce the cost). In particular, the optimal shaping parameters were found to be $\alpha_g^* = 0.67$ and $\alpha_o^* = 0.32$, and the corresponding cost $J^* = 76.57$. The approximation trajectory, obtained using the learned behavior, along with the training trajectory are shown in Figure 3.

For this experiment, we are only optimizing over the shaping parameters for membership functions corresponding to the "go-to-goal" behavior (α_g) and the "avoid-obstacles" behaviors (α_o). Since this optimization only involves two parameters, we can easily compute the cost $J(\alpha_g, \alpha_o)$ over a discrete set of these parameters. Figure 4 depicts the

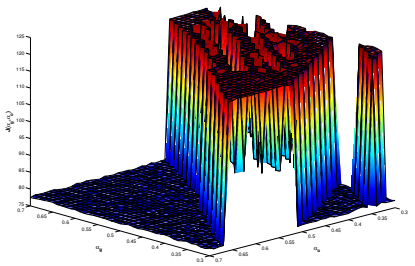


Fig. 4. Depicted is the cost J as a function of the control parameters α_g and α_o. The cost surface is color scaled from low cost (blue) to high cost (red).

approximate cost function parameterized by α_g and α_o. Note that the cost function is highly discontinuous, as we may have expected. Moreover, there are many apparent local extremum to the performance index J. Even so, it is obvious that there are many choices for α_g and α_o that result in significant improvement over other choices (e.g., $J(\alpha_g = 0.67, \alpha_o = 0.32) = 76.57$, while $J(\alpha_g = 0.5, \alpha_o = 0.5) = 119.59$). As mentioned earlier, the performance of the gradient descent algorithm depends on the initial guess of the control vector and the step-size $\gamma^{(p)}$. It is highly recommended that the Armijo step-size be used for the descent algorithm, as this guarantees the algorithm will converge and the cost will be non-increasing. The choice of these parameters may be especially critical for extreme cases such as the one shown in Figure 4.

VI. Conclusions

In this paper, we introduced a constructivist framework for the Learning From Example problem. This framework fits within the control-centric approaches as described in the introduction, but with the fundamental difference that we assume some *a priori* knowledge of possibly relevant control laws. Assuming we have a set of such control laws, new behaviors are learned as a combination of the previous behaviors. Moreover, the system switches between the new, learned control laws in order to approximate the training trajectory. The switching between controllers was time driven, and we presented an algorithm (*inner algorithm*) to calculate the optimal switching times and optimal behavioral combinations in order to minimize a specified performance criterion assuming a fixed number of modes. This is complemented with the *outer algorithm* that determines the number of modes necessary to reproduce the observed trajectory from training example. A small-scale navigation example is discussed in order to highlight the operation of the proposed approach. The initial study of constructivist approach to learning control laws from example seems promising. A natural extension to this approach will be to let the switching between controllers be time-driven, as this may be more relevant to mobile robot applications.

References

[1] F. Delmotte, M. Egerstedt and A. Austin. "Data-Driven Generation of Low-Complexity Control Programs," *International Journal of Hybrid Systems*, Vol. 4(1), pp. 53-72, 2004.

[2] M. Egerstedt, T. Balch, F. Dellaert, F. Delmotte, and Z. Khan. What are the ants doing? vision-based tracking and reconstruction of control programs. *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, Barcelona, April 2005.

[3] T. M. Jochem, D. A. Pomerleau, C. E. Thorpe. Vision-based neural network road and intersection detection and traversal, *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems*, pp. 344-349, 1995

[4] D. Kim, J. Sun, S. M. Oh, J. M. Rehg, and A. Bobick. Traversability Classification Using Unsupervised On-Line Visual Learning for Outdoor Robot Navigation, *IEEE Intl. Conf. on Robotics and Automation*, Orlando, FL, May 2006. , 2005.

[5] J. Sun, T. R. Mehta, D. Wooden, M. Powers, J. Regh, T. Balch, and M. Egerstedt. Learning from Examples in Unstructured, Outdoor Environments, to appear in the *Journal of Field Robotics*.

[6] D. Pomerleau. ALVINN: An autonomous land vehicle in an neural network, *Advances in Neural Information Processing Systems*, pp. 305-313, 1989.

[7] R. Sukthankar, D. Pomerleau, C. Thorpe. A Distributed Tactical Reasoning Framework for Intelligent Vehicles, *Proc. of Intelligent Systems and Manufacturing*, 1997.

[8] A. Guillory, H. Nguyen, T. Balch, C. Isbell. Learning Executable Agent Behaviors from Observation, *Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems*, 2006.

[9] B. Webb. What does robotics offer animal behaviour. *Animal Behaviour*, 60:545558, 2000.

[10] I. Ulrich. I. Nourbakhsh. Appearance-Based Obstacle Detection with Monocular Color Vision, AAAI National Conference on Artificial Intelligence, pp. 866-871, 2000.

[11] T. Balch, F. Dellaert, A. Feldman, A. Guillory, C. Isbell, Z. Khan, S. Pratt, A. Stein, H. Wilde. How A.I. and Multi-Robot Systems Research Will Accelerate Our Understanding of Social Animal Behavior, *Proceedings of the IEEE*, 2006.

[12] J. Michels, A. Saxena, and A. Y. Ng. High Speed Obstacle Avoidance using Monocular Vision and Reinforcement Learning, *International Conference on Machine Learning*, 2005.

[13] J. Rosenblatt. Maximizing Expected Utility for Optimal Action Selection under Uncertainty, *Autonomous Robots*, vol. 9, no. 1, pp. 17-25, 2000.

[14] L. Crawford, and S.S. Sastry. Learning Controllers for Complex Behavioral Systems. *Neural Information Processing Systems Tenth Annual Conference(NIPS 96)*, 1996.

[15] G. Ferrari-Trecate, M. Muselli, D. Liberati and M. Morari. A Clustering Technique for the Identification of Piecewise Affine and Hybrid Systems. *Automatica* 39, pp. 205-217, 2003.

[16] R. Vidal, S. Soatto, Y. Ma and S. Sastry. An Algebraic Geometric Approach to the Identification of a Class of Linear Hybrid Systems. In *Proceedings of IEEE Conference on Decision and Control*, 2003.

[17] J. Bransford, A. L. Brown, and R.R. Cocking. *How People Learn: Brain, Mind, Experience, and School*, Washington: National Academies Press, 2000.

[18] D. Wood. *How Children Think and Learn*, 2nd edition. Oxford: Blackwell Publishers Ltd, 1998.

[19] T.R. Mehta, F. Delmotte and M. Egerstedt. Motion Alphabet Augmentation Based on Past Experience, *IEEE Conference on Decision and Control*, Seville, Spain, Dec. 2005 Optimal Membership

[20] T.R. Mehta and M. Egerstedt. An Optimal Control Approach to Mode Generation in Hybrid Systems, *Nonlinear Analysis: Theory, Methods and Applications*

[21] T.R. Mehta and M. Egerstedt. Optimal Membership Functions for Multi-Modal Control, *American Control Conference*, Minneapolis, Minnesota USA, June 2006.

[22] R.W. Brockett. "Hybrid Models for Motion Control Systems," *Perspectives in Control*, H. Trentelman and J. C. Willems, Eds, Birkh, Boston, pp. 29-54, 1993.

[23] D. Driankov, H. Hellendoorn, and M. Reinfrank. *An Introduction to Fuzzy Control*, Springer, Berlin, 1993.

[24] L. Armijo. Minimization of Functions Having Lipschitz Continuous First-Partial Derivatives. *Pacific Journal of Mathematics*, Vol. 16, ppm. 1-3, 1966.